APPARATUS AND METHOD FOR DIRECT MEMORY ACCESS IN A HUB-BASED MEMORY SYSTEM

TECHNICAL FIELD

This invention relates to computer systems, and, more particularly, to a computer system including a system memory having a memory hub architecture.

BACKGROUND OF THE INVENTION

Computer systems use memory devices, such as dynamic random access memory ("DRAM") devices, to store data that are accessed by a processor. These memory devices are normally used as system memory in a computer system. In a typical computer system, the processor communicates with the system memory through a processor bus and a memory controller. The processor issues a memory request, which includes a memory command, such as a read command, and an address designating the location from which data or instructions are to be read. The memory controller uses the command and address to generate appropriate command signals as well as row and column addresses, which are applied to the system memory. In response to the commands and addresses, data are transferred between the system memory and the processor. The memory controller is often part of a system controller, which also includes bus bridge circuitry for coupling the processor bus to an expansion bus, such as a PCI bus.

Although the operating speed of memory devices has continuously increased, this increase in operating speed has not kept pace with increases in the operating speed of processors. Even slower has been the increase in operating speed of memory controllers coupling processors to memory devices. The relatively slow speed of memory controllers and memory devices limits the data bandwidth between the processor and the memory devices.

In addition to the limited bandwidth between processors and memory devices, the performance of computer systems is also limited by latency problems that

increase the time required to read data from system memory devices. More specifically, when a memory device read command is coupled to a system memory device, such as a synchronous DRAM ("SDRAM") device, the read data are output from the SDRAM device only after a delay of several clock periods. Therefore, although SDRAM devices can synchronously output burst data at a high data rate, the delay in initially providing the data can significantly slow the operating speed of a computer system using such SDRAM devices.

One approach to alleviating the memory latency problem is to use multiple memory devices coupled to the processor through a memory hub. In a memory hub architecture, a system controller or memory controller is coupled over a high speed data link to several memory modules. Typically, the memory modules are coupled in a point-to-point or daisy chain architecture such that the memory modules are connected one to another in series. Thus, the memory controller is coupled to a first memory module over a first high speed data link, with the first memory module connected to a second memory module through a second high speed data link, and the second memory module coupled to a third memory module through a third high speed data link, and so on in a daisy chain fashion.

Each memory module includes a memory hub that is coupled to the corresponding high speed data links and a number of memory devices on the module, with the memory hubs efficiently routing memory requests and responses between the controller and the memory devices over the high speed data links. Computer systems employing this architecture can have a higher bandwidth because a processor can access one memory device while another memory device is responding to a prior memory access. For example, the processor can output write data to one of the memory devices in the system while another memory device in the system is preparing to provide read data to the processor. Moreover, this architecture also provides for easy expansion of the system memory without concern for degradation in signal quality as more memory modules are added, such as occurs in conventional multi-drop bus architectures.

Although computer systems using memory hubs may provide superior performance, they nevertheless may often fail to operate at optimum speeds for a variety of reasons. For example, even though memory hubs can provide computer systems with a greater memory bandwidth, they still suffer from latency problems of the type described above. More specifically, although the processor may communicate with one memory device while another memory device is preparing to transfer data, it is sometimes necessary to receive data from one memory device before the data from another memory device can be used. In the event data must be received from one memory device before data received from another memory device can be used, the intervention of the processor continues to slow the operating speed of such computer systems. Another one of the reasons such computer systems fail to operate at optimum speed is that conventional memory hubs are essentially single channel systems since all control, address and data signals must pass through common memory hub circuitry. As a result, when the memory hub circuitry is busy communicating with one memory device, it is not free to communicate with another memory device.

One technique that has been used in computer systems to overcome the issues with processor intervention in moving data to and from memory as well as the single channel bottleneck is the use of direct memory access (DMA) operations. DMA operations are implemented through the use of DMA controllers included in the computer system which enable data to be moved into and out of memory without the intervention of the system processor. Such DMA operations and DMA controllers are well known in the art, and are often implemented in conventional computer systems. The DMA controller removes the need for the processor to be involved and manages the required data transfers into and out of the system memory. For example, when a DMA supported entity transfers data to the system memory, the DMA controller obtains control of the bus and coordinates the transfer of the data from the DMA supported entity to the system memory, without involvement by the processor. In this manner, latency issues resulting from processor intervention can be avoided during data transfers across the system bus. However, in many

instances, even after data has been transferred to the system memory through a DMA operation, the processor nevertheless must move blocks of the data from one location to another within the system memory. For example, the operating system will direct a DMA operation to transfer data from a mass storage device into the system memory, only to have the processor then move the data again to another location in memory so the data can be used. As a result, the value of having DMA operations is diminished to some degree because the processor ultimately becomes involved by moving data around in memory despite the use of a DMA operation in the data transfer to and from the system memory.

Therefore, there is a need for a computer architecture that provides the advantages of a memory hub architecture and also minimizes the latency problems common in such systems.

SUMMARY OF THE INVENTION

The present invention is directed to a memory hub for a memory module having a DMA engine for performing DMA operations in system memory. The memory hub includes a link interface for receiving memory requests for access to at least one of the memory devices of the system memory, and further including a memory device interface for coupling to the memory devices, the memory device interface coupling memory requests to the memory devices for access to at least one of the memory devices. A switch for selectively coupling the link interface and the memory device interface is further included in the memory hub. Additionally, a direct memory access (DMA) engine is coupled through the switch to the memory device interface to generate memory requests for access to at least one of the memory devices to perform DMA operations.

In an aspect of the present invention, a method is provided for executing memory operations in a computer system having a processor, a system controller coupled to the processor, and a system memory having at least one memory module coupled to the system controller through a memory bus. The method includes writing DMA information to a location in the system memory representing instructions for executing memory

operations in the system memory without processor intervention, obtaining control of the memory bus from the processor and system controller, accessing the location in the system memory to which the DMA information is written, and executing the memory operations represented by the instructions.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computer system according to one example of the invention in which a memory hub is included in each of a plurality of memory modules.

Figure 2 is a block diagram of a memory hub used in the computer system of Figure1.

Figure 3 is a block diagram of a portion of a DMA engine according to an embodiment of the present invention of the memory hub of Figure 2.

Figure 4 is a block diagram of the tag structure according to an embodiment of the present invention used by the DMA engine of Figure 3.

Figure 5 is a flow diagram for operation of a DMA engine of Figure 3 according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present invention are directed to a system memory having a memory hub architecture including direct memory access (DMA) capability to transfer data within the system memory without the intervention of a system processor. Certain details are set forth below to provide a sufficient understanding of the invention. However, it will be clear to one skilled in the art that the invention may be practiced without these particular details. In other instances, well-known circuits, control signals, and timing protocols have not been shown in detail in order to avoid unnecessarily obscuring the invention.

A computer system 100 according to one example of the invention is shown in Figure 1. The computer system 100 includes a processor 104 for performing various computing functions, such as executing specific software to perform specific calculations or tasks. The processor 104 includes a processor bus 106 that normally includes an address bus, a control bus, and a data bus. The processor bus 106 is typically coupled to cache memory 108, which, as previously mentioned, is usually static random access memory ("SRAM"). Finally, the processor bus 106 is coupled to a system controller 110, which is also sometimes referred to as a "North Bridge" or "memory controller."

The system controller 110 serves as a communications path to the processor 104 for a variety of other components. More specifically, the system controller 110 includes a graphics port that is typically coupled to a graphics controller 112, which is, in turn, coupled to a video terminal 114. The system controller 110 is also coupled to one or more input devices 118, such as a keyboard or a mouse, to allow an operator to interface with the computer system 100. Typically, the computer system 100 also includes one or more output devices 120, such as a printer, coupled to the processor 104 through the system controller 110. One or more data storage devices 124 are also typically coupled to the processor 104 through the system controller 110 to allow the processor 104 to store data or retrieve data from internal or external storage media (not shown). Examples of typical storage devices 124 include hard and floppy disks, tape cassettes, and compact disk read-only memories (CD-ROMs).

The system controller 110 includes a memory hub controller 128 that is coupled to several memory modules 130a, 130b, ...130n, which serve as system memory for the computer system 100. The memory modules 130 are preferably coupled to the memory hub controller 128 through a high-speed link 134, which may be an optical or electrical communication path or some other type of communications path. In the event the high-speed link 134 is implemented as an optical communication path, the optical communication path may be in the form of one or more optical fibers, for example. In such case, the memory hub controller 128 and the memory modules will include an optical

input/output port or separate input and output ports coupled to the optical communication path.

The memory modules 130 are shown coupled to the memory hub controller 128 in a point-to-point arrangement in which the high-speed link 134 is formed from coupling together the memory hubs 140 of the memory modules 130. That is, the high speed link 134 is a bi-directional bus that couples the memory hubs 140 in series. Thus, information on the high speed link 134 must travel through the memory hubs 140 of "upstream" memory modules 130 to reach a "downstream" destination. For example, with specific reference to Figure 1, information transmitted from the memory hub controller 128 to the memory hub 140 of the memory module 130c will pass through the memory hubs 140 of the memory modules 130a and 130b. However, it will be understood that other topologies may also be used, such as a coupling arrangement in which each of the memory modules 130 are coupled to the memory hub controller 128 over a high-speed link. A switching topology may also be used in which the memory hub controller 128 is selectively coupled to each of the memory modules 130 through a switch (not shown). Other topologies that may be used will be apparent to one skilled in the art.

As also shown in Figure 1, the memory hub is coupled to four sets of memory devices 148 through a respective bus system 150. Each of the sets includes four memory devices 148 for a total of 20 memory devices 148 for each memory module 130. The bus systems 150 normally include a control bus, an address bus, and a data bus, as known in the art. However, it will be appreciated by those ordinarily skilled in the art that other bus systems, such as a bus system using a shared command/address bus, may also be used without departing from the scope of the present invention. It will be further appreciated that the arrangement of the memory devices 148, and the number of memory devices 148 can be modified without departing from the scope of the present invention. In the example illustrated in Figure 1, the memory devices 148 are synchronous dynamic random access memory ("SDRAM") devices. However, memory devices other than SDRAM devices may, of course, also be used.

An embodiment of a memory hub 200 according to an embodiment of the present invention is shown in Figure 2 that can be substituted for the memory hub 140 of Figure 1. The memory hub 200 is shown in Figure 2 as being coupled to four memory devices 240a-d, which, in the present example are conventional SDRAM devices. In an alternative embodiment, the memory hub 200 is coupled to four different banks of memory devices, rather than merely four different memory devices 240a-d, each bank typically having a plurality of memory devices. However, for the purpose of providing an example, the present description will be with reference to the memory hub 200 coupled to the four memory devices 240a-d. It will be appreciated that the necessary modifications to the memory hub 200 to accommodate multiple banks of memory is within the knowledge of those ordinarily skilled in the art.

Further included in the memory hub 200 are link interfaces 210a-d and 212a-d for coupling the memory module on which the memory hub 200 is located to a first high speed data link 220 and a second high speed data link 222, respectively. As previously discussed with respect to Figure 1, the high speed data links 220, 222 can be implemented using an optical or electrical communication path or some other type of communication path. The link interfaces 210a-d, 212a-d are conventional, and include circuitry used for transferring data, command, and address information to and from the high speed data links 220, 222, as well known, for example, transmitter and receiver logic known in the art. It will be appreciated that those ordinarily skilled in the art have sufficient understanding to modify the link interfaces 210a-d, 212a-d to be used with the specific type of communication path, and that such modifications to the link interfaces 210a-d, 212a-d can be made without departing from the scope of the present invention. For example, in the event the high-speed data link 220, 222 is implemented using an optical communications path, the link interfaces 210a-d, 212a-d will include an optical input/output port and will convert optical signals coupled through the optical communications path into electrical signals.

The link interfaces 210a-d, 212a-d are coupled to the a switch 260 through a plurality of bus and signal lines, represented by busses 214. The busses 214 are conventional, and include a write data bus and a read data bus, although a single bi-directional data bus may alternatively be provided to couple data in both directions through the link interfaces 210a-d, 212a-d. It will be appreciated by those ordinarily skilled in the art that the busses 214 are provided by way of example, and that the busses 214 may include fewer or greater signal lines, such as further including a request line and a snoop line, which can be used for maintaining cache coherency.

The link interfaces 210a-d, 212a-d include circuitry that allow the memory hub 140 to be connected in the system memory in a variety of configurations. For example, the multi-drop arrangement, can be implemented by coupling each memory module to the memory hub controller 128 through either the link interfaces 210a-d or 212a-d. Alternatively, a point-to-point, or daisy chain configuration, as shown in Figure 1, can be implemented by coupling the memory modules in series. For example, the link interfaces 210a-d can be used to couple a first memory module and the link interfaces 212a-d can be used to couple a second memory module. The memory module coupled to a processor, or system controller, will be coupled thereto through one set of the link interfaces and further coupled to another memory module through the other set of link interfaces. In one embodiment of the present invention, the memory hub 200 of a memory module is coupled to the processor in a point-to-point arrangement in which there are no other devices coupled to the connection between the processor 104 and the memory hub 200. This type of interconnection provides better signal coupling between the processor 104 and the memory hub 200 for several reasons, including relatively low capacitance, relatively few line discontinuities to reflect signals and relatively short signal paths.

The switch 260 is further coupled to four memory interfaces 270a-d which are, in turn, coupled to the system memory devices 240a-d, respectively. By providing a separate and independent memory interface 270a-d for each system memory device 240a-d, respectively, the memory hub 200 avoids bus or memory bank conflicts that typically occur

with single channel memory architectures. The switch 260 is coupled to each memory interface through a plurality of bus and signal lines, represented by busses 274. The busses 274 include a write data bus, a read data bus, and a request line. However, it will be understood that a single bi-directional data bus may alternatively be used instead of a separate write data bus and read data bus. Moreover, the busses 274 can include a greater or lesser number of signal lines than those previously described.

In an embodiment of the present invention, each memory interface 270a-d is specially adapted to the system memory devices 240a-d to which it is coupled. More specifically, each memory interface 270a-d is specially adapted to provide and receive the specific signals received and generated, respectively, by the system memory device 240a-d to which it is coupled. Also, the memory interfaces 270a-d are capable of operating with system memory devices 240a-d operating at different clock frequencies. As a result, the memory interfaces 270a-d isolate the processor 104 from changes that may occur at the interface between the memory hub 230 and memory devices 240a-d coupled to the memory hub 200, and it provides a more controlled environment to which the memory devices 240a-d may interface.

The switch 260 coupling the link interfaces 210a-d, 212a-d and the memory interfaces 270a-d can be any of a variety of conventional or hereinafter developed switches. For example, the switch 260 may be a cross-bar switch that can simultaneously couple link interfaces 210a-d, 212a-d and the memory interfaces 270a-d to each other in a variety of arrangements. The switch 260 can also be a set of multiplexers that do not provide the same level of connectivity as a cross-bar switch but nevertheless can couple the some or all of the link interfaces 210a-d, 212a-d to each of the memory interfaces 270a-d. The switch 260 may also includes arbitration logic (not shown) to determine which memory accesses should receive priority over other memory accesses. Bus arbitration performing this function is well known to one skilled in the art.

With further reference to Figure 2, each of the memory interfaces 270a-d includes a respective memory controller 280, a respective write buffer 282, and a respective

cache memory unit 284. The memory controller 280 performs the same functions as a conventional memory controller by providing control, address and data signals to the system memory device 240a-d to which it is coupled and receiving data signals from the system memory device 240a-d to which it is coupled. The write buffer 282 and the cache memory unit 284 include the normal components of a buffer and cache memory, including a tag memory, a data memory, a comparator, and the like, as is well known in the art. The memory devices used in the write buffer 282 and the cache memory unit 284 may be either DRAM devices, static random access memory ("SRAM") devices, other types of memory devices, or a combination of all three. Furthermore, any or all of these memory devices as well as the other components used in the cache memory unit 284 may be either embedded or stand-alone devices.

The write buffer 282 in each memory interface 270a-d is used to store write requests while a read request is being serviced. In a such a system, the processor 104 can issue a write request to a system memory device 240a-d even if the memory device to which the write request is directed is busy servicing a prior write or read request. Using this approach, memory requests can be serviced out of order since an earlier write request can be stored in the write buffer 282 while a subsequent read request is being serviced. The ability to buffer write requests to allow a read request to be serviced can greatly reduce memory read latency since read requests can be given first priority regardless of their chronological order. For example, a series of write requests interspersed with read requests can be stored in the write buffer 282 to allow the read requests to be serviced in a pipelined manner followed by servicing the stored write requests in a pipelined manner. As a result, lengthy settling times between coupling write request to the memory devices 270a-d and subsequently coupling read request to the memory devices 270a-d for alternating write and read requests can be avoided.

The use of the cache memory unit 284 in each memory interface 270a-d allows the processor 104 to receive data responsive to a read command directed to a respective system memory device 240a-d without waiting for the memory device 240a-d to

provide such data in the event that the data was recently read from or written to that memory device 240a-d. The cache memory unit 284 thus reduces the read latency of the system memory devices 240a-d to maximize the memory bandwidth of the computer system. Similarly, the processor 104 can store write data in the cache memory unit 284 and then perform other functions while the memory controller 280 in the same memory interface 270a-d transfers the write data from the cache memory unit 284 to the system memory device 240a-d to which it is coupled.

Further included in the memory hub 200 is a DMA engine 286 coupled to the switch 260 through a bus 288 which enables the memory hub 200 to move blocks of data from one location in the system memory to another location in the system memory without intervention from the processor 104. The bus 288 includes a plurality of conventional bus lines and signal lines, such as address, control, data busses, and the like, for handling data transfers in the system memory. As will be described in more detail below, the DMA engine 286 is able to read a link list in the system memory to execute the DMA memory operations without processor intervention, thus, freeing the processor 104 and the bandwidth limited system bus from executing the memory operations. The DMA engine 286 is preferably an embedded circuit in the memory hub 200. However, including a separate DMA device coupled to the memory hub 200 is also within the scope of the present invention. Additionally, the DMA engine 286 can include circuitry to accommodate DMA operations on multiple channels. Such multiple channel DMA engines are well known in the art and can be implemented using conventional technologies.

In an embodiment of the present invention, the processor 104 writes a list of instructions in the system memory for the DMA engine 286 to execute. The instructions include information used by the DMA engine 286 to perform the DMA operation, such as starting address of the block to move, ending address or count, destination address, the address of the next command block, and the like. The DMA engine 286 will execute a series of continuous commands and then jump to the next command list if directed to do so. The DMA engine 286 is programmed through a data structure that exists in one or more

memory spaces. The data structure consists of some number of command blocks that provide information necessary to perform data transfer operations in the system memory. Each of the command blocks can be linked through a series of address pointers to form a linked list. The address of the first command block in the linked list is programmed through the I/O space. The DMA engine 286 is instructed to fetch and execute the first command block through the I/O space command register. After performing the requested data operation, an address pointer in the first command block is used to point the DMA engine 286 to the next command block. An address pointer in each successive command block is used to fetch and execute the next command block, forming a linked list. Each command block in the linked list is executed until a NULL pointer is encountered. An example of a NULL pointer is defined as an address consisting of all 1's. Upon detecting the NULL pointer, command block execution will halt, and a status bit will be set, indicating the command stream has terminated. Completion status can be contained in an I/O register in the memory hub 200. Additionally, a start flag can also be used to indicate that the DMA engine 286 has already begun executing the DMA operation. Other status bits can indicate if the command stream has terminated normally with no errors, or abnormally due to errors. The status information may optionally generate an interrupt to the host.

In alternative embodiments of the present invention, the DMA engine 286 can also be used for running diagnostics in the system. Known good data patterns can be loaded in memory of the memory hub 200, or known good system memory, and be used to test the system memory. A more detailed description of this type of application is provided in commonly assigned, co-pending U.S. Patent Application No. _____, entitled SYSTEM AND METHOD FOR ON-BOARD DIAGNOSTICS OF MEMORY MODULES, filed on [Filing Date], which is incorporated herein by reference.

Figure 3 is a block diagram illustrating portions of a DMA engine 300 and Figure 4 is a block diagram illustrating a linked command list table 400 according to embodiments of the present invention. The DMA engine 300 can be substituted for the

DMA engine 286 of the memory hub 200 (Figure 2). It will be appreciated that Figure 3 is merely a representation of the DMA engine 300, and those ordinarily skilled in the art are provided sufficient description herein in order to practice the present invention. However, it will be further appreciated that alternative DMA engines can also be used without departing from the scope of the present invention. The DMA engine 300 includes five registers: an address register 310, a destination address register 311, a control register 312, a next register 314, and a count register 316, to control DMA operations.

In operation, at the beginning of a block transfer, the starting address for the block is loaded into the address register 310. Additionally, a destination address of the location to which data is to be moved is loaded into the destination address register 311, and the length of the block is loaded into the count register 316. The control register 312 contains information relevant to the transfer, such as a bit indicating whether the address register 310 is to be incremented or decremented after each data item is transferred. In the present example, every time a data item is transferred by the DMA engine 300, the count register 316 is decremented and the address register 310 is incremented. Additionally, the destination address register 311 is incremented (or decremented, depending on the control settings). When the value of the count register 316 reaches zero, the block transfer has been completed. At this time, the value in the next register 314 is checked. If it points to a valid location in the system memory, the values contained in that object are loaded into the registers 310, 312, 314, and 316. A next block data transfer then begins automatically. However, if a NULL value, as previously described, is present in the next register 314, the DMA operation is complete.

The linked command list table 400 shown in Figure 4 contains a plurality of link entries 402, 404, and 406, each of which contains the information necessary to reload registers 310, 312, 314, and 316. The link entries 402, 404, and 406 are stored in the system memory, as previously discussed, and are linked together by pointers corresponding to the next register 314. In Figure 4, three link entries 402, 404, and 406 are shown. These link entries, plus an initial transfer defined by writing values directly into the registers 310,

312, 314, and 316 of the DMA engine 300, define a single DMA transfer having four separate parts. The value NEXT, contained in the next register 314, points to the first link entry 402. The first link entry 402 points to the next link entry 404 in the linked command, which in turn points to the final link entry 406. The final link entry 406 contains the NULL value as a pointer, indicating that it is the last link entry of a DMA command list. The NULL value is a reserved pointer value which does not point to a valid memory location, and is interpreted by the DMA engine 300 as a pointer to nothing. It will be appreciated that the link entries 402, 404, 406 are provided by way of example, and modifications thereto, such as including greater or fewer fields of information than that shown in Figure 4, can be made without departing from the scope of the present invention.

Figure 5 is a flow diagram 500 illustrating the control flow used by the DMA engine 300 (Figure 3) to make a series of consecutive block transfers. At a step 502, the DMA registers 310, 312, 314, and 316 are loaded with the appropriate values for the first data transfer. At this time, either before or after loading the registers directly, all of the information necessary for the link entries for this transfer must be loaded into the linked command list table 400 (Figure 4). Loading of the registers is at the command of the processor 104 (Figure 1) and loading of the linked command list 400 in the system memory is accomplished by the processor 104 as well.

At a step 504, one data item is transferred, and at a step 506, the value in the count register 316 is decremented to indicate that one data item has been transferred. The step 506 includes simultaneously incrementing or decrementing the value of the address register 310, depending upon the desired direction as set in the control register 312. At a step 508, the count value is checked to determine whether the count is complete. In one embodiment of the present invention, determination of whether the count is complete is accomplished by checking a carry out bit (not shown) from the count register 316. In the event the count value indicates that the data transfer is not complete, control returns to the step 504. However, if the count value in the count register 316 is equal to zero, control passes to a step 510, where the value in the next register 314 is tested to see if it is equal to

the NULL value, as previously described. If a NULL value is not present, at a step 512 the next tag is loaded into the registers 310, 312, 314, and 316 in the DMA controller 300 from the linked command list table 400, and control returns to the step 504. Once the last link entry has been used, at a step 514 an indication is made to the processor 104 that the transfer is complete.

It will be appreciated by those ordinarily skilled in the art that the DMA engine 300 implements a "scatter-gather" capability for use in the system memory. When a large block of data is to be read into nonconsecutive blocks of memory, the processor 104 allocates the memory and sets up the linked command list table 400 through the DMA engine 300. A DMA transfer is then initiated, and the DMA engine 300 handles the entire transfer until it is completed. A similar technique can be used for gathering scattered blocks of data within the system memory in order to write them to consecutive blocks of memory. The processor 104 determines which blocks are to be written moved within the system memory, and their order, and sets up the linked command list table 400 through the DMA engine 300. A DMA transfer is then initiated, and is handled completely by the DMA engine 300 until it is completed. Since the linked command list table 400 is stored in the system memory, it is possible to keep several linked lists, for example, for each channel supported by the DMA engine 300. Moreover, since the linked command list table 400 is stored in the system memory, the only limit on the number of separate transfers which may be linked into one larger transfer for a channel is the number of remaining free memory locations within the system memory.

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.